Gradient Descent

CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader and Chris Tanner

Learning the coefficients

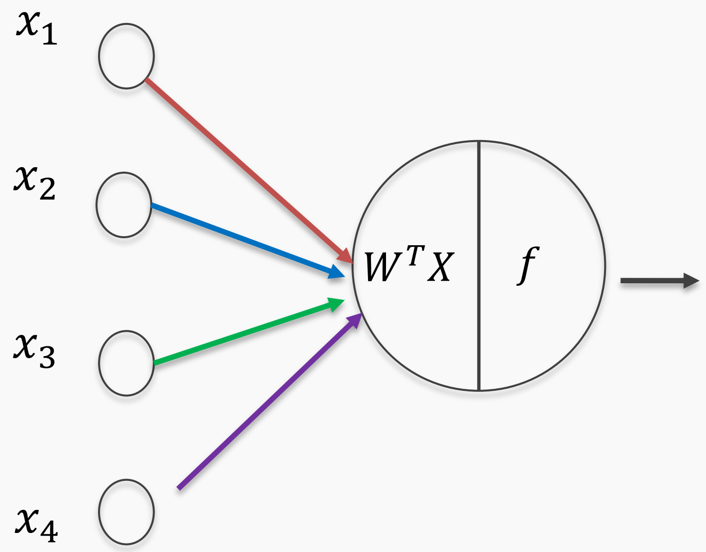
Start with single neuron

Classification

$f(X)=\frac{1}{1+e^{-W^TX}}$

Regression

$f(X) = W^TX$



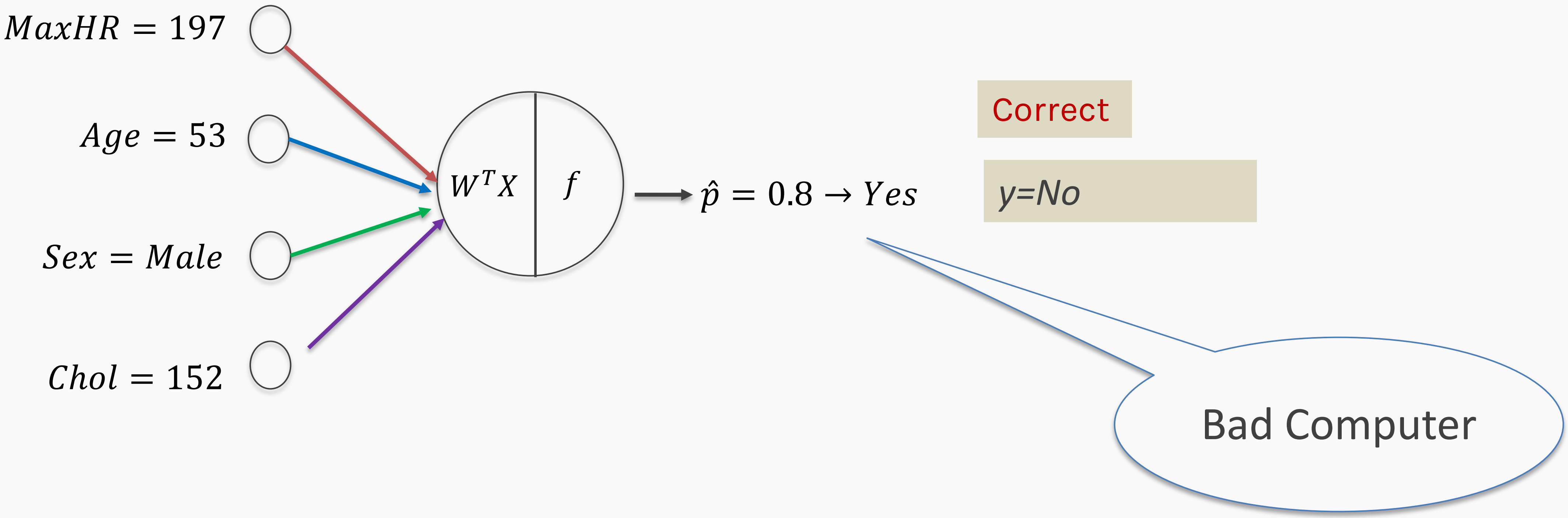
$$Y = f(W_0 + W_1x_1 + W_2x_2 + W_3x_3 + W_4x_4)$$

Intercept or **Bias**

Coefficients or **Weights**

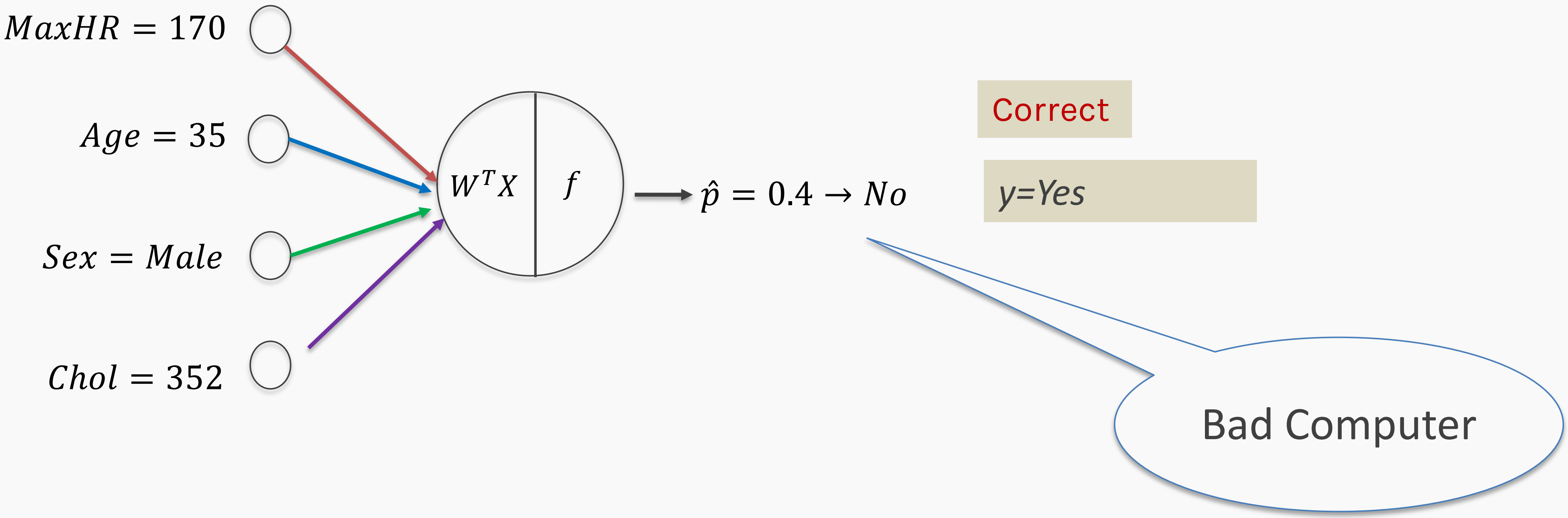
But what is the idea?

Start with all randomly selected weights. Most likely it will perform horribly. For example, in our heart data, the model will be giving us the wrong answer.



But what is the idea?

Start with all randomly selected weights. Most likely it will perform horribly. For example, in our heart data, the model will be giving us the wrong answer.



But what is the idea?

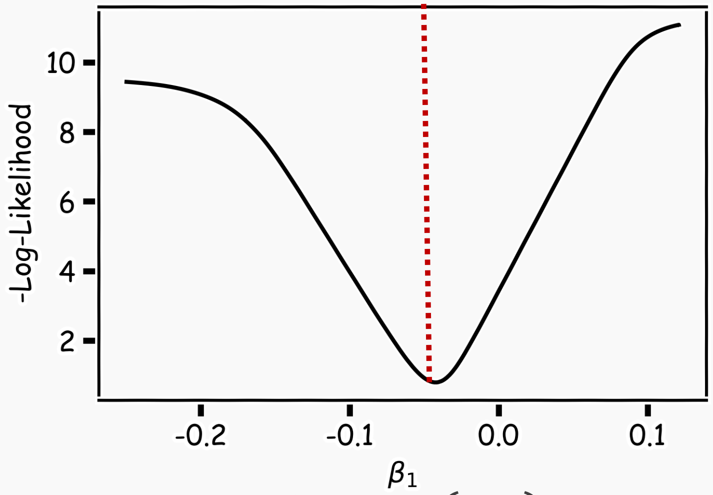
- **Loss Function:** Takes all of these results and averages them and tells us how bad or good the computer or those weights are.
- Telling the computer how **bad** or **good** is, does not help.
- You want to tell it how to change those weights so it gets better.

Loss function: $\mathcal{L}(w_0, w_1, w_2, w_3, w_4)$

For now let's only consider a single weight, $\mathcal{L}(w_1)$

Minimizing the Loss function

Ideally we want to know the value of w_1 that gives the minimal $\mathcal{L}(W)$

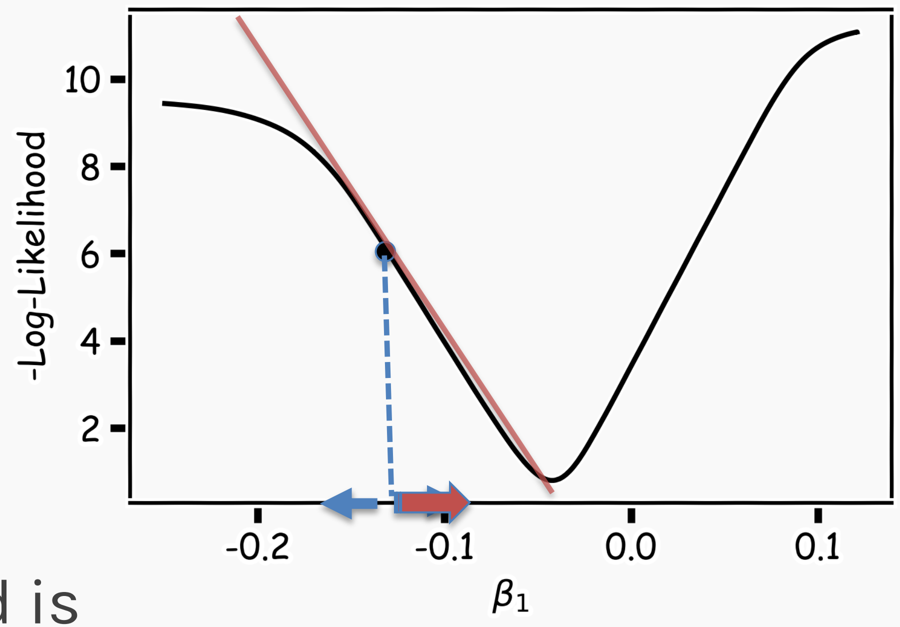


To find the optimal point of a function $\mathcal{L}(W)$

$$\frac{d\mathcal{L}(W)}{dW} = 0 \qquad \text{solve for } W^* = \text{argmin}_W \mathcal{L}(W)$$

And find the W that satisfies that equation. **Sometimes** there is no explicit solution for that.

Estimate of the regression coefficients: gradient descent



A more flexible method is

- Start from a random point
  1. Determine which direction to go to reduce the loss (left or right)
  2. Compute the slope of the function at this point and step to the right if slope is negative or step to the left if slope is positive
  3. Goto to #1

Minimization of the Loss Function

Question: What is the mathematical function that describes the slope?

Derivative

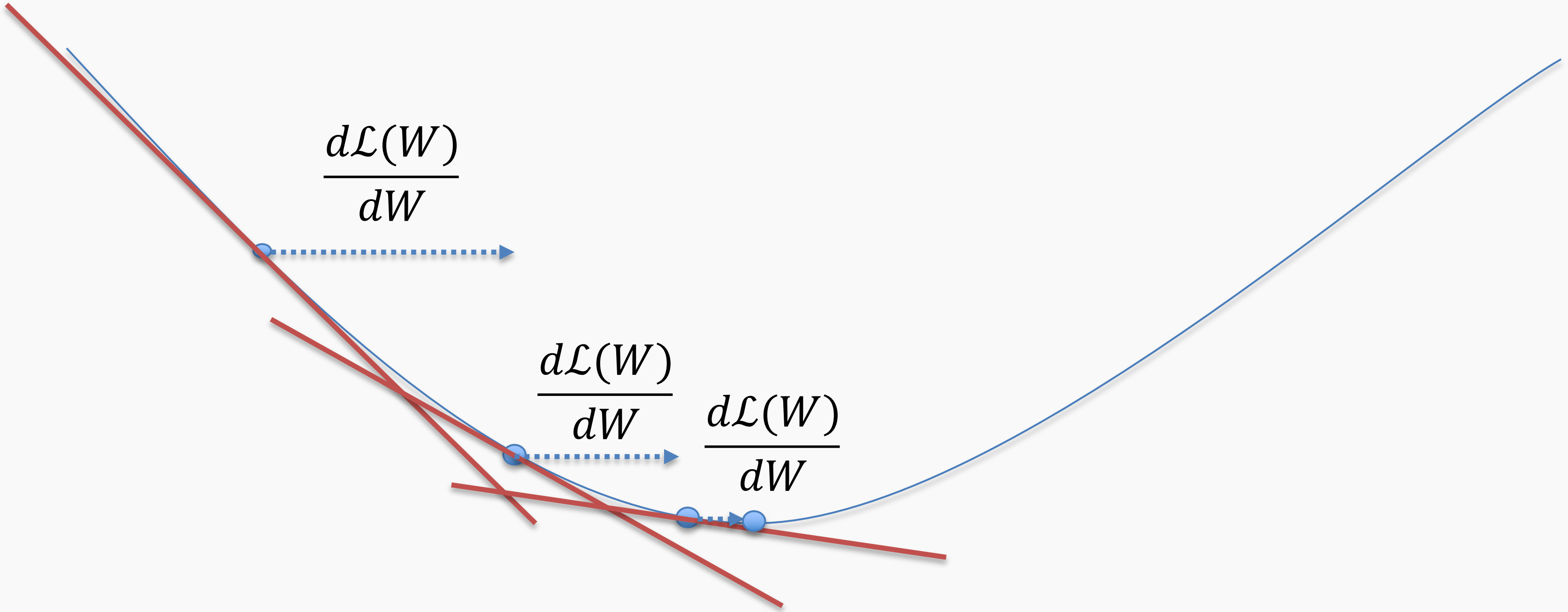
Question: How do we generalize this to more than one predictor?

Take the derivative with respect to each coefficient and do the same sequentially

Question: What do you think is a good approach for telling the model how to change (what is the step size) to become better?

Gradient Descent (cont.)

If the step is proportional to the slope then you avoid overshooting the minimum. How?



Let's play the Pavlos game

We know that we want to go in the opposite direction of the derivative and we know we want to be making a step proportional to the derivative.

Making a step means:

$$w^{new} = w^{old} + step$$

Opposite direction of the derivative means:

$$w^{new} = w^{old} - \eta \frac{d\mathcal{L}}{dw}$$

Step size is proportional to derivative

Change to more conventional notation:

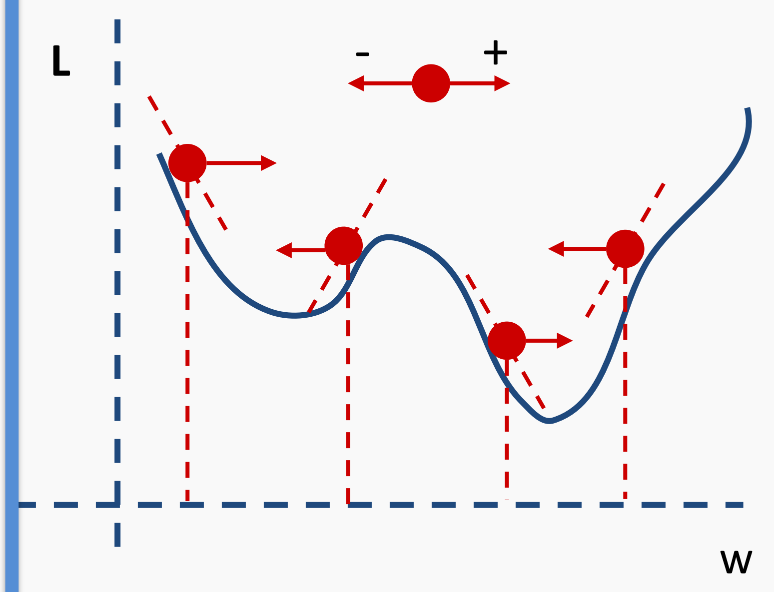
$$w^{(i+1)} = w^{(i)} - \eta \frac{d\mathcal{L}}{dw}$$

Learning Rate

Gradient Descent

- Algorithm for optimization of first order to finding a minimum of a function.
- It is an iterative method.
- L is decreasing much faster in the direction of the negative derivative.
- The learning rate is controlled by the magnitude of η.

$$w^{(i+1)} = w^{(i)} - \eta \frac{d\mathcal{L}}{dw}$$



IS YOUR CHILD TEXTING ABOUT

DEEP LEARNING?

brb - backprogation right back
lmao - layering multiple activation optimizers
stfu - support Theano for users!
smh - sponsor my hardware(GPUs)
rofl - ReLU optimization for logistic regression
lol - linear overfitted lasso
btw - but tensorflow works
idc - I do CNNs
omg - oh my gradients!
gdi - gradient descent intensity
rn - recurrent neuralnet
fml - forever machine learning!